Accepted Manuscript

Optimizing the inventorying and distribution of chemical fluids: An innovative nested column generation approach.

Mariana E. Coccola , Carlos A. Mendez , Rodolfo G. Dondo

PII: S0098-1354(18)30284-9
DOI: https://doi.org/10.1016/j.compchemeng.2018.08.004
Reference: CACE 6176

To appear in: *Computers and Chemical Engineering*

Received date: 5 April 2018
Revised date: 26 July 2018
Accepted date: 3 August 2018

Please cite this article as: Mariana E. Coccola , Carlos A. Mendez , Rodolfo G. Dondo , Optimizing the inventorying and distribution of chemical fluids: An innovative nested column generation approach., *Computers and Chemical Engineering* (2018), doi: https://doi.org/10.1016/j.compchemeng.2018.08.004

Highlights

- **Optimal integration of inventorying and distribution of fluid chemicals.**
- **Nested column generation embedded within an incomplete branch-and-price algorithm.**
- **Routes and delivery patterns separately computed by the algorithm.**
- **Several real-world examples efficiently solved with moderate computation effort.**

Optimizing the inventorying and distribution of chemical fluids: An innovative nested column generation approach.

Mariana E. Coccola, Carlos A. Mendez, Rodolfo G. Dondo*
INTEC (Universidad Nacional del Litoral - CONICET)
Güemes 3450 (3000) Santa Fe- Argentina,
**Corresponding author: rdondo@santafe-conicet.gov.ar*

Abstract

Vendor-Managed-Inventory is a successful business practices based on the cooperation between a supplier and its customers in which demand and inventory information from the customers are shared with the supplier. This practice is gaining popularity in the chemical industry and relies on the inventory-routing-problem, which integrates inventory management, vehicle routing, and delivery scheduling decisions. This one is a difficult combinatorial optimization problem both theoretically and practically. However, because of the large expenses involved in distribution and inventorying of chemical products, it is attractive to make use of optimization tools for exploiting as many degrees of freedom as possible with the goal of minimizing both distribution and inventorying costs. Consequently, we propose a nested column generation algorithm for solving an inventorying and distribution problem that models the delivery of several chemicals fluids. The approach is building on a column generation & incomplete branch-and-price algorithm in which for each delivery route, the delivery patterns of fluids are also determined by column generation. We detail the implementation and provide computational results for realistic test instances.

Keywords*: multi-commodity; nested column generation; incomplete branch-and-price; multi-compartment vehicles; inventory-routing-problem.*

Acronyms

B&P	Branch-and-price
CG	Columns generation
DP	Delivery pattern
EWO	Enterprise-wide optimization
GUB	Global upper bound
IRP	Inventory routing problem
LIN	Liquid nitrogen
LLB	Local lower bound
LOX	Liquid oxygen
PGP	Pattern generation problem
RGP	Routes generation problem
RMP	Restricted master problem
RPGMP	Restricted pattern-generation master problem
SDVRP	Split-delivery vehicle routing problem
VMI	Vendor-managed-inventory

Nomenclature

Subscripts

a	minimum-distance arcs interconnecting suppliers and customers
c	vehicle's compartments
i,i',j	suppliers or customers
k	products
n	nodes
p	patterns
r	routes
t,t'	periods of the planning horizon

Sets

A	minimum-distance arcs interconnecting suppliers and customers
C	vehicle's compartments
$Current_n$	next node to explore in the branching tree
$Exception_r$	excluded columns in the current node
Fx_{nij}^{0}	pairs *(i,j)* fixed to zero in the node n
Fx_{nij}^{1}	pairs *(i,j)* fixed to one in the node n
Fx_{ij}	pairs *(i,j)* fixed in the current node
I^-	customers
I^+	production plants
K	products
N	nodes of the branching tree
$Newnode_n$	new nodes in the tree after branching
P_r	delivery patterns associated to route r
R_t	feasible routes of period t
T	periods of the planning horizon
$Wainting_n$	nodes in the branching tree without exploring

Binary variables

S_{ij}	variable for sequencing locations i and j along a route
X_{rt}	variable for selecting route r of period t
X_{rt}^{kc}	variable for allocating product k to compartment c on the truck traveling route r of period t
Y_i	variable used to determine that the site i belongs to the route designed by a routes-generator problem
Z_{ck}	variable for allocating product k on compartment c

Continuous variables

C_r	cost of route r
C_p	cost of pattern p
D_i	distance travelled to reach customer i
Q_{ikc}	quantity of product k to pick-up(deliver) from(to) source(sink) site i on compartment c
T_i	time spent to reach customer i
TV	total routing time
X_{rt}^{kpc}	variable weighting the contribution of pattern p to the load of product k on compartment c by route r of period t

X_p^r	variable for weighting the contribution of pattern *p* on route *r*
Λ_{ikrt}	quantity of product *k* picked(delivered) from(to) site *i* by route *r* of period *t*

Parameters

a_{irt}	binary parameter stating that route *r* of period *t* visits location *i*
a_i	earliest service time at customer i
α_{rt}^{ikpc}	quantity of product *k* picked(delivered) on compartment *c* from(to) site *i* by route *r* of period *t* according to pattern *p*
b_i	latest service time at node i
$best_{ri}$	saving the facilities *i* visited in the route *r* for the best solution found
$bestX_{rt}$	saving the routes *r* selected for the period *t* in the best solution found
bestBound	for selecting the node with the lower bound more promising
bestFound	best solution found in the branching tree
$bound_n$	lower bound *(LLB)* of the node *n*
c_{ij}	traveling cost of arc *i - j*
c_{rt}	cost of route *r* of period *t*
c_p	cost of pattern *p*
cf_v	fixed vehicle utilization cost
cv_v	distance unit travelling cost
d_{ij}	distance between locations *i* and *j*
d_{itk}	demand of product *k* by customer *i* during the period *t*
GUB	optimal solution of the integer RMP
i_{ik}^0	initial inventory of product *k* on location *i*
i_{ik}^{Max}	maximum storage capacity of product *k* on location *i*
i_{ik}^{Min}	minimum inventory level or safety-stock for product *k* on location *i*
LLB	optimal solution of the linear RMP
M_D, M_T	upper bounds for travelled distance *(D)* and travel time *(T)* varibles
p_{itk}	production of product *k* by plant *i* during the period *t*
q_{itk}	upper bound on the quantity of product *k* to pickup/deliver from/to source/sink node *i* during period *t*
q_c	cargo capacity of compartment *c*
st_i	service time on customer (plant) *i*
$t^{\max}$	maximum vehicle routing time
t_{ij}	traveling time between locations *i* and *j*
v_i	minimum number of visits to customer (plant) *i*
π_i^-	price associated to the visit to customer *i*
π_i^+	price associated to the visit to plant *i*
π_{itk-}^{Max}	price associated to constraint on the maximum storage capacity of product *k* on customer *i*
π_{itk+}^{Max}	price associated to constraint on the maximum storage capacity of product *k* on plant *i*
π_{itk-}^{Min}	price associated to constraint on the minimum inventory level of product *k* on customer *i*
π_{itk+}^{Min}	price associated to constraint on the minimum inventory level of product *k* on plant *i*
π_{ikt}	price associated to covering constraint of the pattern generation master problem

1. Introduction

The scale of the chemical industry is global, being logistics a crucial area of this type of industry, because raw materials sources, production facilities and consumer markets are globally distributed. Also, due to the increasing pressure for reducing costs, inventories and ecological footprint, and in order to remain competitive in the global marketplace, Enterprise-wide optimization (EWO) has become a major goal of the chemical industry (Grossmann, 2012). In this way, fluctuating demand, seasonal imbalances of raw materials and products flows, as well as expensive transportation and inventorying motivate a dynamic and integrated management of logistic activities. Vendor-managed-inventory (VMI) is a successful business practice based on the cooperation between a supplier and its customers in which demand and inventory information from the customers are shared with the supplier (Desaulniers et al, 2016). In VMI, customer inventories are replenished by the vendor using monitoring and forecasting in a way that each product-inventory on each customer must be replenished so as to never fall under the safety level. Under the modality, illustrated in Figure 1, the supplier is responsible for managing the inventory level of the customers, deciding when and how much to deliver to each one.

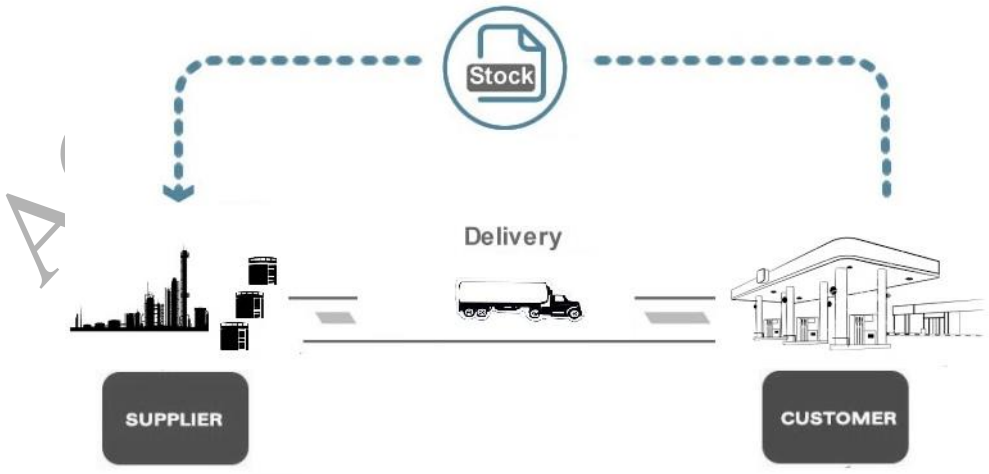

Figure 1: VMI relies on monitoring customer stocks to allow the supplier to decide when and how much products to deliver

VMI modality allows the supplier to better integrate the visits to several customers and thus smooth its production, inventory, and distribution efforts (Adulyasak et al. 2015). In order to operate along such a strategy, the supplier should solve an inventory-routing problem (IRP), which combines over a multi-period time-horizon inventory management, vehicle routing, and delivery scheduling decisions. In such a context, the supplier has to take three simultaneous decisions: (i) when to serve a given customer, (ii) how much to deliver to this customer and (iii) how to combine customers into the design of the vehicle-routes. The aim is to find the optimal trade-off between vehicle routing costs and inventory holding costs, such that total costs are minimized. The problem is rooted in a classic paper from Bell et al. (1983) about distribution and inventorying of gas but it was later extended beyond that scope. The IRP has been researched considering different replenishment strategies but the two most popular replenishment policies are the order-up-to-level and the maximum-level policies (Desaulniers et al, 2016). In the order-up-to-level policy each delivery must fill the customer inventory to its maximum capacity, so once the decision to visit a customer is taken, the quantity to be delivered is computed as the difference between its maximum capacity and its current inventory level. This policy has been proposed by Dror et al. (1985). In the maximum-level policy, any quantity can be delivered as long as the maximum capacity is not exceeded. The ML policy encompasses the order-up-to-level alternative and is more flexible, but also more complex given the additional decision to be taken. The IRP is a difficult combinatorial optimization problem, both theoretically and practically, and different methods for solving it have been proposed. See for example the paper by Dong et al (2017) that presents interesting algorithms for solving the problem.

Several applications of the IRP have been recorded in the literature and many of them arise in maritime logistics, i.e. in ship routing and inventory management. Reviews about the subject are provided by Ronen (1993); Christiansen at al. (2004); Christiansen et al. (2007) and Christiansen et al. (2013). Non-maritime applications of the IRP arise in a large variety of industries, including the distribution of gas by tanker trucks (Bard et al. 1998; Campbell and Savelsbergh 2004; Golden et al, 1984; Trudeau and Dror, 1992), distribution of automobile components (Alegre at al., 2007; Blumenfeld et al. 1985; Stacey et al., 2007), distribution of perishable items (Federgruen and Zipkin 1984; Federgruen at al., 1986), transportation of groceries (Custódio and Oliveira 2006; Gaur and Fisher 2004; Mercer and Tao 1996), distribution of cement (Christiansen et al. 2011), distribution of fuel (Popovic et al., 2012), of blood (Hemmelmayr et al. 2009), of livestock (Oppen et al., 2010), of waste vegetable oil collection (Aksen et al. 2012), of crude oil (Shen et al., 2011) and production and distribution of industrial gases (You et al., 2014; Marchetti et al., 2014; Zamarripa et al., 2016, Singh et al., 2015).

This work is the second one on a research line aimed at the optimal integration of inventorying and distribution of fluid chemicals. It follows a previous one by Cóccola et al. (2017) focused on optimizing the order-based-resupply of chemical fluids. The current work aims at modelling and optimizing the inventorying and distribution of several chemical fluids by multi-compartment trucks. Its main contributions are the following:

1. A set-partitioning model for optimizing the multi-period inventorying and distribution of several chemical fluids is presented. We consider here the case of bulk delivery by multi-compartments trucks.

2. An incomplete branch-and-price algorithm based on a nested column generation procedure is codified to solve relatively large instances of the above described problem.
3. Computational experiments on instances featuring different characteristics are performed to test the capability of the algorithm for providing effective and efficient solutions.

The remainder of this paper is organized as follows: Section 2 describes the methodology for bulk delivery of multiple fluid products by multi-compartments trucks. The mathematical formulation developed to represent this resupply modality is presented in Section 3. Then, this model is reformulated on section 4 in order to develop an incomplete branch and price algorithm based on the separated generation of routes and delivery patterns by a nested column generation algorithm. Numerical experiments over a series of instances featuring different characteristics are presented in Section 5. Finally, the concluding remarks follow in Section 6.

2. Inventory and distribution of multiple fluids

To address the inventorying and distribution problematic illustrated in Figure 2, let's consider some customers spread over a given geographical area. Each customer consumes several fluid products, which are sourced from plants producing them. Customers are equipped with a multi-commodity storage and similarly, each plant has a multi-commodity storage from which the products can be pumped out. Customer demands and plants production volumes over each period of the whole planning horizon are known data and the following issues must be addressed by the planner:

1. When to resupply a given customer?
2. What quantity of products must be supplied to each visited customer?

3. Which clients to deliver on each period and from which plant?

4. How many vehicles must be used on each period?

5. How to fill each vehicle for servicing the assigned clients?

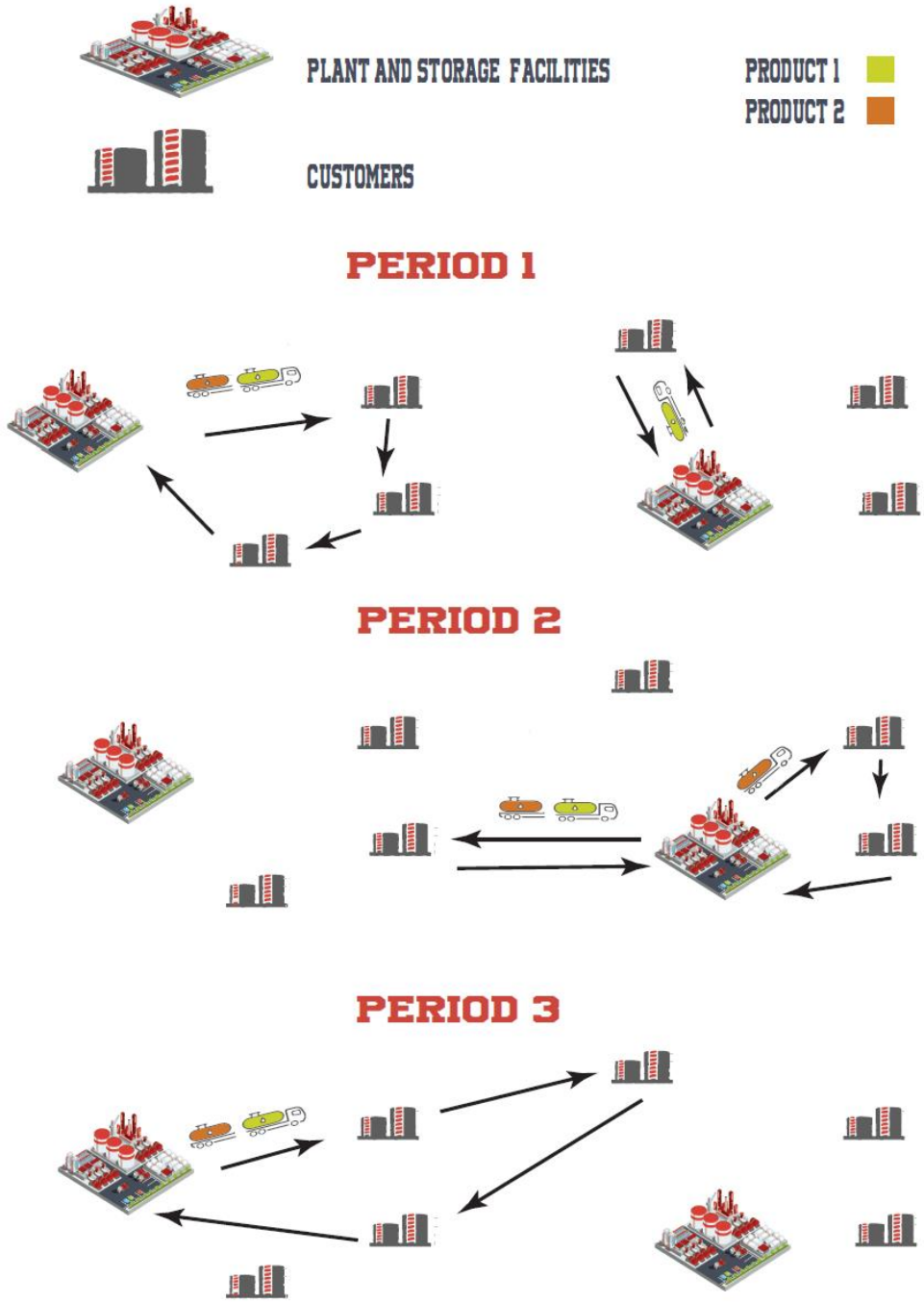

Figure 2: Illustration of the inventorying and distribution problematic

Considering the complexity and the dimension of the addressed problems, which typically involve several plants, dozens of clients and several products; it is practically impossible to optimally solve a monolithic mathematical model for finding solutions useful as answers to the above stated questions. Therefore, this paper presents a decomposition approach developed to find near optimal solutions for real world problems with moderate computational effort. The solution strategy is based on a nested column generation algorithm which decouples routing decisions from delivering decisions.

3. Model formulation

In order to formally define the problem previously defined let us consider a set of suppliers, denoted by set $I^+ = \{i_{1+}, i_{2+}, \ldots i_{n+}\}$. Each supplier $i \in I^+$ produces a known quantity p_{itk} of commodity $k \in K$ during the period t of the planning horizon $T = \{t_1, t_2, \ldots, t_t\}$. By using a homogeneous fleet of multi-compartment vehicles, each one with $|C|$ compartments of capacity q_c, the supplier supply a set of customers $I^- = \{i_{1-}, i_{2-}, \ldots, i_{n-}\}$. Each customer $i \in I^-$ consumes a known quantity d_{itk} of commodity $k \in K$ during the period t of the planning horizon T. Both the suppliers $i \in I^+$ and the customers $i \in I^-$ have multi-compartment storage tanks with a storage capacity ($I_{ik}^{Max} - I_{ik}^{Min}$), where I_{ik}^{Max} is the maximum storage capacity and I_{ik}^{Min} is the minimum operative capacity or "safety stock" below which the inventory of the product must never fall at the end of any time-period. In addition, for each product $k \in K$, a known initial inventory I_{ik}^{0} at the start of the first time-period is available both on suppliers $i \in I^+$ and on customers $i \in I^-$. All facilities are connected through a network represented by a graph G ($I^+ \cup I^-$, A), where A is the network of minimum-distance arcs interconnecting suppliers and customers. Associated to every arc $a \in A$, there are a distance-based

traveling cost c_{ij} and a travel time t_{ij}. Costs incurred for operating the vehicles are a fixed utilization cost cf_v and a distance-unit travelling cost cv_v. The visiting time of a vehicle on a location $i \in I^+ \cup I$ is denoted by st_i. The problem aims at finding the optimal set of delivery routes during each period $t \in T$ such that, for each product $k \in K$, the storage maximum-capacity is respected and no stock-outs occur on each customer and each plant. A vehicle route is considered feasible if it satisfies the vehicle compartments-capacity constraint and a maximum time-length $t^{\max}$. The objective is to minimize the sum of vehicle travelling costs and inventory holding costs at both the suppliers and the customers. Frequently, holding costs are considered negligible with respect to routing costs when non-perishable products, like fuels, are distributed. In this paper, we research this option.

In order to formulate this problematic as a mixed integer-linear program (MILP), let us define R_t as the set of all feasible replenishment routes corresponding to the period $t \in T$. Each route $r \in R_t$ is characterized by a cost c_{rt} given by the sum of the costs of the arcs travelled by the vehicle plus the fixed vehicle utilization cost. In addition, a binary parameter a_{irt} is used to indicate whether route $r \in R_t$ visits ($a_{irt} = 1$) or not ($a_{irt} = 0$) the location $i \in I^+ \cup I$ during the period $t \in T$. For computing the optimal solution, two variables are used: a binary decision variable X_{rt} valuing 1 if route $r \in R_t$ is selected, and a positive variable Λ_{ikrt} that takes a value equal to the quantity of product $k \in K$ loaded (unloaded) on plant $i \in I^+$ (customer $i \in I$) by the route $r \in R_t$. Then, the problem can be stated as the following MILP:

Mininimize

$$\sum_{t \in T} \sum_{r \in R_t} c_{rt} X_{rt} \qquad (1)$$

subject to:

$$\sum_{t\in T}\sum_{r\in R_t} a_{irt} X_{rt} \geq v_i \qquad \forall i \in I^+ \cup I^- \tag{2}$$

$$\left\{ \begin{array}{l} I_{ik}^0 - \sum\limits_{t'\in T: t'<t} d_{itk} + \sum\limits_{t'\in T: t'\leq t} \sum\limits_{r\in R_t} \Lambda_{ikrt} \leq I_{ik}^{Max} \\ I_{ik}^0 - \sum\limits_{t'\in T: t'\leq t} d_{itk} + \sum\limits_{t'\in T: t'\leq t} \sum\limits_{r\in R_t} \Lambda_{ikrt} \geq I_{ik}^{Min} \end{array} \right\} \qquad \forall i \in I^-, t \in T, k \in K \tag{3.a}$$

(3.b)

$$\left\{ \begin{array}{l} I_{ik}^0 + \sum\limits_{t'\in T: t'\leq t} p_{itk} - \sum\limits_{t'\in T: t'\leq t} \sum\limits_{r\in R_t} \Lambda_{ikrt} \leq I_{ik}^{Max} \\ I_{ik}^0 + \sum\limits_{t'\in T: t'<t} p_{itk} - \sum\limits_{t'\in T: t'\leq t} \sum\limits_{r\in R_t} \Lambda_{ikrt} \geq I_{ik}^{Min} \end{array} \right\} \qquad \forall i \in I^+, t \in T, k \in K \tag{4.a}$$

(4.b)

$X_{rt} \in \{0, 1\}$

$\Lambda_{ikrt} \geq 0$

The objective function (1) minimizes the total traveling costs; storage costs are assumed to be negligible. Constraints (2) state that the minimum number of visits to a plant/customer node over the whole planning horizon T, given by the parameter v_i, must be covered. Constraints (3) are inventory constraints for each product k on every customer $i \in I$ during each period t. They state that the initial product-inventory I_{ik}^0 plus the quantities of the product delivered to the customer minus the total consumption of the product up to the current time period must be larger than the minimum allowed stock I_{ik}^{Min} (at the end of period $t \in T$) and smaller than the maximum storage capacity I_{ik}^{Max} (at the start of each period $t \in T$). Eqs. (4) are similar to eqs. (3) but they are applied to suppliers defined by the set $i \in I^+$. The simultaneous determination of optimal decisions X_{rt} and optimal quantities Λ_{ikrt} for each feasible route may be quite cumbersome. In this way, the formulation (1) - (4) may be solved just for small scale instances with a realistically enumerable quantity of feasible routes. The resolution of large instances of the problem leads to the use of decomposition procedures because the

simultaneous determination of optimal values for binary variables X_{rt} and continuous variables Λ_{ikrt} becomes much harder.

4. Reformulation and solution approach

The reformulation of the model (1)-(4) represents the continuous variables Λ_{ikrt} associated with a given route as a weighted sum of "extreme" delivery pattern (DP) as it was proposed by Desaulniers (2010) for the split-delivery vehicle routing problem with time windows (SDVRP). In this way, two different concepts are introduced; routes and delivery patterns. A route starts at a supply plant, visits a sequence of customers and ends on the same plant. Each route must be completed in a single time-period $t \in T$. A delivery pattern (DP) associated to a route $r \in R_t$ specifies both the quantity of products taken from the supplier and the quantity delivered to each customer along the route. As stated by Desaulniers (2010), just "extreme" DPs are needed to generate any DP through their convex combination. For the single-commodity single-compartment case, in an extreme DP there is at most one customer per route for which a fraction of its demand is partially covered (called a split-delivery). The other customers of such a pattern receive a zero quantity (called a zero-delivery) or their demands are totally covered (called a full-delivery). Any feasible pattern would be obtained as a convex combination of extreme DPs involving full deliveries and zero-deliveries to the visited customers. It is worth noting that the problem here considered is more complex than the SDVRP because of the delivery and inventorying of multiple products through a number of vehicles with several compartments. Consequently, the definition of extreme DP is more complicated. In this way, the quantity of product k to pick up (deliver) from (to) plant (customer) location i during period t by route r can be computed as follows:

$$\Lambda_{ikrt} = \sum_{p \in P_r} \left(\sum_{c \in C} \alpha_{rt}^{ikpc} X_{rt}^{kpc} \right) \qquad \forall t \in T, r \in R_t, i \in I^+ \cup I^-, k \in K \qquad (5.a)$$

$$\sum_{p \in P_r} X_{rt}^{kpc} = X_{rt}^{kc} \qquad \forall t \in T, r \in R_t, k \in K, c \in C \qquad (5.b)$$

$$\sum_{k \in K} X_{rt}^{kc} = X_{rt} \qquad \forall t \in T, r \in R_t, c \in C \qquad (5.c)$$

where $\alpha^{ikpc}{}_{rt}$ is the quantity of product k to pick up (deliver) from(to) source(sink) location i in compartment c by the extreme cargo-pattern p associated to route r of period t; $X^{kpc}{}_{rt}$ is a continuous variable bounded by the interval [0, 1] for weighting the contribution of the extreme DP to the optimal DP just in case the route $r \in R_t$ belongs to the optimal solution ($X_{rt} = 1$) and X_{rt}^{kc} is a binary value for allocating a single product in each compartment of the used vehicle. Eq. (5.a) computes the optimal quantity Λ_{ikrt} as a weighted sum of quantities $\alpha^{ikpc}{}_{rt}$ on all active DPs and on all compartments. Eq. (5.b) forces the sum of weights to take the value of binary variable X_{rt}^{kc} and if route r is part of the optimal solution, i.e. $X_{rt} = 1$, eq. (5.c) forces that just one X_{rt}^{kc} variable can take value 1. In other words, this equation is formulated for allocating just one product per compartment in order to avoid products mixing. By introducing eqs. (5) into eqs. (2), (3) and (4) we can now reformulate the model (1)-(4) as follows:

Mininimize

$$\sum_{t \in T} \sum_{r \in R_t} c_{rt} \left(\sum_{p \in P_r} \frac{1}{|C|} \sum_{c \in C} \sum_{k \in K} X_{rt}^{kpc} \right) \qquad (6)$$

subject to:

$$\sum_{t \in T} \sum_{r \in R_t} a_{irt} \left(\sum_{p \in P_r} \frac{1}{|C|} \sum_{c \in C} \sum_{k \in K} X_{rt}^{kpc} \right) \geq v_i \qquad \forall i \in I^+ \cup I^- \qquad (7)$$

$$\left\{\begin{array}{l}\sum_{t'\in T:t'\le t}\sum_{r\in R_t}\sum_{c\in C}\sum_{p\in P_{rk}}\alpha_{rt}^{ikpc}X_{rt}^{kpc}\le I_{ik}^{Max}-I_{ik}^{0}+\sum_{t'\in T:t'<t}d_{itk}\\ \sum_{t'\in T:t'\le t}\sum_{r\in R_t}\sum_{c\in C}\sum_{p\in P_r}\alpha_{rt}^{ikpc}X_{rt}^{kpc}\ge I_{ik}^{Min}-I_{ik}^{0}+\sum_{t'\in T:t'\le t}d_{itk}\end{array}\right\}\quad \forall t\in T, i\in I^{-}, k\in K \qquad (8.a)\ (8.b)$$

$$\left\{\begin{array}{l}\sum_{t'\in T:t'\le t}\sum_{r\in R_t}\sum_{c\in C}\sum_{p\in P_r}\alpha_{rt}^{ikpc}X_{rt}^{kpc}\ge -I_{ik}^{Max}+I_{ik}^{0}+\sum_{t'\in T:t'\le t}p_{itk}\\ \sum_{t'\in T:t'\le t}\sum_{r\in R_t}\sum_{c\in C}\sum_{p\in P_r}\alpha_{rt}^{ikpc}X_{rt}^{kpc}\le -I_{ik}^{Min}+I_{ik}^{0}+\sum_{t'\in T:t'<t}p_{itk}\end{array}\right\}\quad \forall t\in T, i\in I^{+}, k\in K \qquad (9.a)\ (9.b)$$

$$\sum_{p\in P_r}X_{rt}^{kpc}=X_{rt}^{kc} \qquad \forall t\in T, r\in R_t, k\in K, c\in C \qquad (10.a)$$

$$\sum_{k\in K}X_{rt}^{kc}=X_{rt} \qquad \forall t\in T, r\in R_t, c\in C \qquad (10.b)$$

$$X_{rt}\in\{0,1\}$$

$$X_{rt}^{kc}\in\{0,1\}$$

Eqs. (6) to (9) are the eqs. (1) to (4) reformulated according to eqs. (5.a). Moreover, eqs. (5.b)-(5.c) must be introduced as eqs. (10.a)-(10.b) into the reformulation because the best DP for each route *r* of period *t* would arise as a sum on all compartments of the convex combination of DPs included in the dynamic set P_r. Eq. (10.a) forces the sum of weights $X^{kpc}{}_{rt}$ on P_r to take the value of variable $X^{kc}{}_{rt}$. In turn, eq. (10.b) allocates a single product to a given vehicle compartment just in case X_{rt} = 1. When column generation is used to solve model (6)-(10), each iteration must incorporate new routes and associated DPs to sets R_t and P_r. This implies that new constraints (10) must be added as well.

The formulation (6)-(10) has a decomposable structure abled to be exploited by the column generation (CG) paradigm. CG is an iterative method usually employed to solve routing problems involving covering constraints like eq. (2). CG is carried out in two phases: a slave problem which generates feasible routes, also named columns, and a master problem, where all columns are brought together to find the optimal set of

routes. Unfortunately, this method cannot be straightforwardly extended to the researched problem because of the complexity of simultaneously computing routes and DPs. Consequently, a more complex algorithm aiming at separately computing replenishment routes and their associated DPs was developed in this paper. The algorithm is based on decomposition into three levels. On the bottom level the procedure computes the extreme DPs for each route generated on the intermediate level problem. A route and its DPs together form the so called "cargo-routes" that are coordinated on the top level master problem. Since some no-generated cargo-routes may exist but they may not be present in the current pool, to find better solutions, the missing columns must be generated after branching. Consequently, to find the optimal (or a better) integer solution, the procedure must be embedded into a branch-and-bound algorithm.

4.1 Restricted master problem (RMP)

CG is a decomposition procedure that solves at each iteration both a master problem restricted to a subset of columns (restricted master problem or RMP) and several sub-problems, also known as pricing problems. The procedure starts with a RMP that contains a small number of cargo-routes. For incorporating new columns to the RPM, one subproblem per period $t \in T$ and per supplier $i \in I^+$ is solved. This means that the routes and DPs referenced by these columns are computed by solving the respective pricing problems for each period and for each supplier. For every new route, associated DP must also be generated and new constraints (10.a)-(10.b) need to be added as well. This is a drawback because the number of feasible routes exponentially increases on the number of visited customers. To overcome this difficulty, we define the RMP by eqs. (6) to (9) and drop constraints (10.a)-(10.b) as is proposed by Hennig et al. (2012). Note

that this pair of constraints is indexed by the elements of sets R_t and consequently, the number of constraints of the RMP will grow accordingly. In this way, after dropping these constraints, duals from equations (7) to (9) are used to generate routes (and DPs). Constraints (10) are, very likely, unsatisfied by the solution to the linear RMP but they must be enforced just by the integer master problem aimed at finding the optimal routes and weights of patterns associated to these routes.

After finding the optimal solution for the RMP, the dual variables values π_i^+ and π_i^- from constraints (7) as well as π^{Max}_{itk-}, π^{Min}_{itk-}, π^{Max}_{itk+} and π^{Min}_{itk+} from constraints (8) and (9) are passed to the pricing problems in order to produce more profitable cargo-routes. At each iteration, the linear problem defined by eqs. (6)-(9) is solved on the current set of columns. Afterwards, the new routes and associated DP generated by the pricing problems are added to the RMP. The iterative procedure continues as far as the optimal solution to the linear problem cannot be improved with the addition of another cargo-route. This happens when any pricing problem cannot return a cargo-route combination with a negative reduced cost.

4.2. Mid-level pricing problem or routes generation problem (RGP)

Once the RMP is solved by the simplex algorithm or by any other LP algorithm, dual values associated to the problem constraints are used to define the objective function of the subproblem. This one aims at identifying new routes with negative reduced cost with respect to the current dual variable values. In order to generate useful routes, the CG procedure pivot on each time-period $t \in T$ and on each plant $i \in I^+$ to solve the following problems:

$\forall i' \in I^+, t \in T$, *solve:*

Minimize

$$\left(C_r + C_p\right) - \left(Y_{i'}\pi_{i'}^{+} + \sum_{t\in T}\sum_{k\in K}\left(\pi_{i'tk+}^{Min} - \pi_{i'tk+}^{Max}\right)\Lambda_{i'krt} \right) - \sum_{i\in I^-}\left(Y_i\pi_i^{-} + \sum_{t\in T}\sum_{k\in K}\left(\pi_{itk-}^{Min} - \pi_{itk-}^{Max}\right)\Lambda_{ikrt} \right) \quad (11)$$

subject to:

$$D_j \geq d_{i'j} \qquad \forall j \in I^- \quad (12)$$

$$\left\{ \begin{array}{c} D_j \geq D_i + d_{ij} - M_D\left(1 - S_{ij}\right) \\ D_i \geq D_j + d_{ji} - M_D S_{ij} - M_D\left(2 - Y_i - Y_j\right) \end{array} \right\} \qquad \forall (i,j) \in I^- : i < j \quad (13.a)\ (13.b)$$

$$C_r \geq cf_v + cv_v\left(D_j + d_{ji'}\right) \qquad \forall j \in I^- \quad (14)$$

$$T_j \geq t_{i'j} \qquad \forall j \in I^- \quad (15)$$

$$\left\{ \begin{array}{c} T_j \geq T_i + st_i + t_{ij} - M_T\left(1 - S_{ij}\right) \\ T_i \geq T_j + st_j + t_{ji} - M_T S_{ij} - M_T\left(2 - Y_i - Y_j\right) \end{array} \right\} \qquad \forall (i,j) \in I^- : i < j \quad (16.a)\ (16.b)$$

$$TV \geq T_j + st_j + t_{ji'} \qquad \forall j \in I^- \quad (17)$$

$$\left\{ \begin{array}{c} T_i \geq a_i \\ b_i \geq T_i \end{array} \right\} \qquad \forall i \in I^- \quad (18.a)\ (18.b)$$

$$\begin{array}{l} Y_{i'} = 1 \\ Y_j = 0 \end{array} \qquad \forall j \in I^+ : j \neq i' \quad (19.a)\ (19.b)$$

$$\sum_{i\in I^-}\sum_{k\in K}\Lambda_{ikrt} \leq \sum_{c\in C} q_c \quad (20)$$

$$\Lambda_{ikrt} \leq Y_i \sum_{c\in C} q_c \qquad \forall i \in I^- \quad (21)$$

$$C_p \geq c_p \sum_{i\in I^-}\sum_{k\in K}\Lambda_{ikrt} \quad (22)$$

$$Y_i \in \{0,1\}$$

The objective function (11) is defined as the sum of the cost of the computed route C_r and the computed pattern C_p minus the prices associated to locations visited along the route minus the prices associated to the upper and lower bounds on the quantity of

products available during each period both on the pivot plant and on the visited customers. Since upper and lower bounds cannot be simultaneously active constraints, duals π^{Max}_{itk-}, π^{Min}_{itk-} (or π^{Max}_{itk+} and π^{Min}_{itk+}) cannot simultaneously have nonzero values. Eq. (12) computes the minimum distance travelled to reach customer $j \in I$ from the pivot plant $i' \in I^+$. Eqs. (13.a)-(13.b) fix the accumulated distance travelled by the tanker-truck up to each visited site. I.e. if customers i and j, with $i < j$, are allocated to the route ($Y_i = Y_j = 1$), the visiting ordering for both sites is determined by the value of the sequencing variable S_{ij}. If customer i is visited before j ($S_{ij} = 1$), according to (13.a), the travelled distance up to the customer j (D_j) must be larger than D_i by at least d_{ij}. In case node j is visited earlier, ($S_{ij} = 0$), the reverse statement holds and eq. (13.b) becomes active. M_D is an upper bound for variables D_i. Eq. (14) computes the total routing cost C_r by converting the total travelled distance on the travelling cost and by adding the fixed vehicle utilization cost. Eqs. (15), (16) and (17) are constraints similar to eqs. (12), (13) and (14) but in this case, these constraints define the minimum time to visit a customer, the visiting time to each customer along the route and the total routing time *TV,* respectively. Eqs. (18) state time-windows constraints. Constraints (19) fix the pivot plant to allocate the vehicle that fulfills the generated route. Vehicle-capacity constraints are stated by eq. (20) while eq. (21) states that no cargo can be delivered to a non-visited customer. Finally, constraint (22) compute the cost C_p associated to the only computed DP.

Note that the routes generation problem computes the minimum-cost cargo-route and the objective (11) depends both on visiting-decision variables Y_i and the positive quantities Λ_{ikrt}. This, in turn, implies that the same route but with a different DP may be generated in several iterations thus complicating the convergence rate and reducing the algorithmic efficiency of the CG procedure. In order to avoid this drawback, we decided

to decouple the routing decisions from the design of DP by exploiting the key idea proposed by Desaulniers (2010), which consists in the use of a few extreme DPs to generate any other feasible DP. So, for a given a route, all extreme patterns would be generated by a lower level pattern generation procedure. Although neglecting pattern-costs implies that the integer RMP may provide many solutions with the same objective function value and different delivery patterns; the nested CG procedure would not be affected by the existence of these solutions.

Whereas in the SDVRP a convex combination of only two extreme DP is sufficient for generating any other DP, for this problem, many DPs may be needed to explicitly list them. So, to avoid the cumbersome enumeration of all extreme DPs of a given route the procedure uses CG again to generate them. In that sense, this decomposition scheme leads to manageability and flexibility. However, the algorithmic structure gets a little more complicated since it nests two CG algorithms.

4.3. Lowest level pattern generation problem

Quantities Λ_{ikrt} are continuous variables that can be computed as a convex combination of extreme DPs. They might be directly generated according to the idea proposed by Desaulniers (2010) but the situation in this problem is considerably more complex because the multi-commodity nature of the problem and the multi-compartment characteristic of the vehicles lead to a combinatory number of extreme DPs. To avoid enumerating them, extreme DPs associated to a given route $r \in R_t$ are again generated by using CG at a lower level of the procedure. This problem is stated as follows:

$\forall t \in T, r \in R_t, solve:$

Restricted pattern generation master problem (RPGMP):

Minimize:

$$\sum_{p \in P_r} c_p X_p^r \tag{23}$$

subject to:

$$\frac{1}{q_{itk}} \sum_{p \in P_r} \left(\sum_{c \in C} Q_{ikc} X_p^r \right) = 1 \qquad \forall i \in I^- : Y_i = 1, k \in K \tag{24}$$

$$0 \le X_p^r \le 1$$

Eq. (23) minimizes the cost of generating DPs associated to the route $r \in R_t$ and eq. (24) states that the maximum deliverable quantity q_{itk} (see appendix) to each customer must be fulfilled at least by one pattern in order to create the "full delivery" of product $k \in K$ to customer $i \in I$ during the period $t \in T$. This constraint was formulated in order to obtain any DP as a convex combination of generated DP. For computing the cargo-route cost, if C_p on all patterns is negligible with respect to C_r, the pattern cost is set to 1 (C_p = 1) with the purpose of minimizing the number of generated DPs and erasing C_p from eq. (11). The *RPGMP* is initialized by a set of single-customer single-product patterns delivering the quantity q_{itk} from the plant to each one of the customers included in the associated route plus a zero-delivery pattern.

Slave pattern generation problem (PGP):

Patterns are dynamically generated by minimizing the objective (25), subject to constraints (26) to (28):

Minimize:

$$c_p - \frac{1}{q_{itk}} \sum_{\substack{i \in I^- : \\ : Y_i = 1}} \sum_{k \in K} \pi_{ikt} \left(\sum_{c \in C} Q_{ikc} \right) \tag{25}$$

subject to:

$$\sum_{k \in K} Z_{ck} \le 1 \qquad \forall c \in C \tag{26}$$

$$\sum_{\substack{i \in I^- : \\ :Y_i = 1}} Q_{ikc} \leq Z_{ck} q_c \qquad \forall c \in C, k \in K \tag{27}$$

$$\sum_{c \in C} Q_{ikc} \leq q_{itk} \qquad \forall i \in I^-, k \in K \tag{28}$$

Eq. (25) is the reduced cost of the generated DP. Eq. (26) states that each compartment of the vehicle, if used, must be allocated just to a single product while eq. (27) is a capacity constraint on the quantity of product loaded to each compartment of the truck. Eq. (28) states that the total quantity of product *k* delivered to customer $i \in I^-$ during period $t \in T$ must not exceed the maximum deliverable quantity q_{ikt}. The PGP runs until no more DPs with negative reduced cost can be generated. After that, quantities α_{rt}^{ikpc} are respectively computed for visited customers and for the supply plant as follows:

$$\alpha_{rt}^{ikpc} = Q_{ikc} \qquad \forall i \in I^- : Y_i = 1, p \in P_r . k \in K, c \in C \tag{29.a}$$

$$\alpha_{rt}^{ikpc} = \sum_{\substack{j \in I^- : \\ :Y_j = 1}} Q_{jkc} \qquad \forall i \in I^+ : Y_i = 1, p \in P_r k \in K, c \in C \tag{29.b}$$

It is worth noting that several options to feed columns to the RMP (6)-(9) may arise. If we want to provide just the minimum number of extreme DP able to produce any other DP, the model defined by eqs. (23)-(24) with $X_p^r \in \{0, 1\}$ must be solved. If we want to avoid the resolution of this problem, all generated DPs associated to a route must be supplied to the RMP. The best option depends on the instance to be solved and should be determined by numerical testing.

4.4. Improving the solution by branching

At the root node of the branch-and-price tree, the linear relaxation of model (6)-(9) is solved. When negative reduced-cost columns, i.e. cargo-routes, are found, they are

added to the RMP which is solved again to start a new iteration. Otherwise, when sub-problems cannot provide at least a cargo-route with negative reduced costs, the CG process stops. Since some non-generated columns pricing favorably may exist but they may not be present in the current RMP, to find better solutions, the missing columns can be generated after branching. Successive linear relaxations in the tree are obtained by adding branching decisions. Since the problem presents a high degree of symmetry (i.e. there are many solutions with different variable-values and the same objective function value) and a given route can be useful in several time periods, the selection of a branching rule is of utmost importance. According to Savelsbergh and Sol (1998), it is good to utilize a branching rule focusing on assignment decisions rather than on routing decisions because assignment decisions constitute higher level decisions and have a greater impact on the structure of the solution. Based on this idea, it is natural to develop the following branching scheme: when the solution is not integer, the procedure selects a customer $i \in I^-$ and a plant $i \in I^+$ and creates two subspaces; on the first one the customer must be replenished from this plant and on the other one the customer must not be replenished from this plant. The idea is similar to that one proposed by Savelsbergh and Sol (1998) which selects a vehicle and a customer to create two sub-spaces but in our case, the vehicle is replaced by a plant $i \in I^+$. When solving the mid-level pricing problem in a given subspace, the first restriction is satisfied by forcing customer $i \in I^-$ to be replenished from $i \in I^+$. The second branching restriction requires a constraint stating that whenever $i \in I^+$ is on the tour, this tour cannot visit $i \in I^-$. Successive branching constraints are straightforwardly applied. To enforce branching constraints in the RMP of a branch-and-price node, it is necessary to exclude columns from the solution space of the branch-and-price node by setting the associated variable bounds to zero. Moreover, the branching rule above explained must also be incorporated

at the pricing level. In order to improve the global upper bound along the search-tree, feasible integer solutions may be recomputed either by exact or heuristic procedures.

4.5. Algorithm setup

The algorithm, illustrated by Figure 3, has been coded in GAMS 23.6.2 and embeds the nested CG within a branch-and-bound procedure that branches according to the rule proposed in section 4.4.

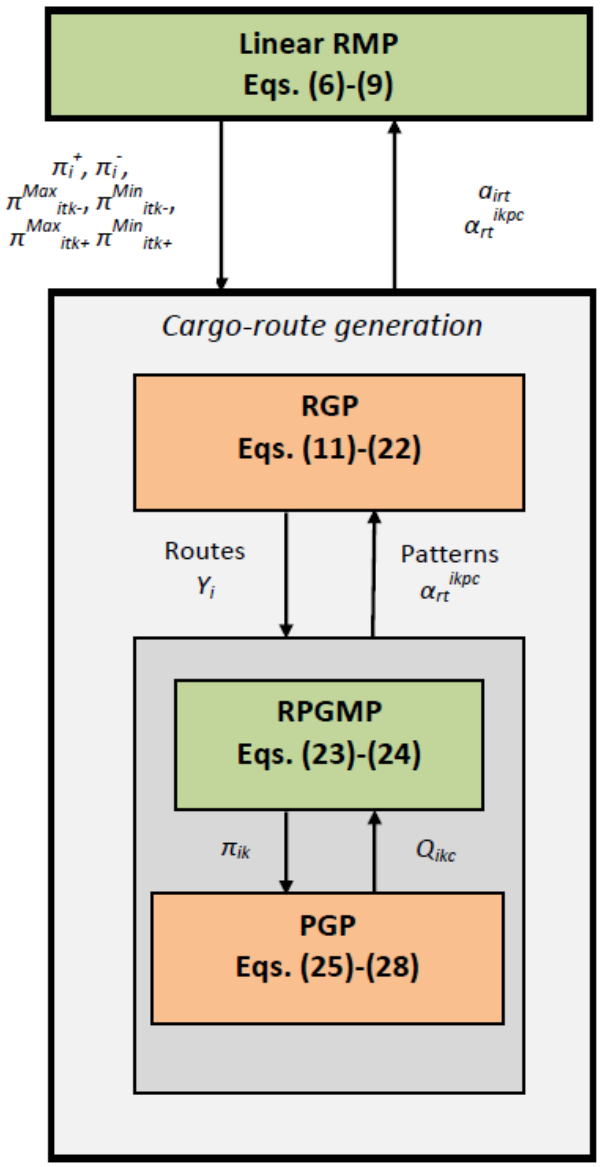

Figure 3: Sketch of the nested cargo-routes generation procedure

The problem presents a high degree of symmetry that implies that a route generated for a given time-period may be useful in other periods. So, in order to avoid generating the same route through different slave problems, we maximize the decoupling of routes generation from patterns generation. In this way, the procedure first solves a heuristic RMP defined by eqs. (6)-(7) in the root node of the branch-and-price tree and load the columns generated in such a way for all periods $t \in T$. After a given number of iterations, the procedure switches to the exact RMP to generate the remaining columns corresponding to each period through the corresponding slave problems. In no-root nodes the procedure is similar but the generated routes are loaded just in periods that fulfil branching constraints. Since the algorithm collects several routes per master-slave iteration via the `SolnPool` procedure (CPLEX Solver manual, 2012), a solutions filter aimed at deleting suboptimal solutions visiting the same customers via non-optimal paths was incorporated into the procedure. Afterwards, for each filtered route, the CG procedure aimed at generating DPs is run. After the resolution of the routing pricing problem, the RMP is fed with just the best routes and their associated DPs.

The algorithm is used in heuristic mode although, in principle, it may be used as an exact algorithm able to locate the optimal solution, if enough computational power and time is available. The procedure starts with a feasible solution used to solve the first RMP. At each iteration, the dual variables associated to the RMP constraints are passed to the mid-level problems to generate the minimum reduced-cost cargo-route. To find all extreme DPs associated to a route, the pattern generation problem summarized in section 4.3 is solved. A route and its associated DPs are then incorporated to the pool of columns of the master problem and the procedure repeats itself until not a cargo-route combination with a negative reduced cost can be found by the mid-level problem(s). To initialize the algorithm, for each time period $t \in T$, feasible cargo-routes $i - j - i$, starting

from any plant $i \in I^+$ and going to any customer $j \in J$ are generated. Single-product DPs delivering the quantity q_{itk}, for each product $k \in K$, are associated to each one of these single customer routes. From this initial cargo-routes package, the linear RMP can be computed to start the master-slave recursion.

The branching mechanism is sketched in the left column of Figure 4. The right column provides brief explanations of the row aligned sentences. The algorithm uses CPLEX 11 as the MIP solver for generating both routes and DPs and for computing upper and lower bounds. Since branch-and-price is an enumeration algorithm enhanced by fathoming based on bound comparisons, the strongest bounds should be the best ones but the mechanism can work with any bound. Nevertheless, the best upper bound might need the resolution of an integer RMP while the best node lower-bound can be obtained by solving the relaxed RMP just after no more profitable cargo-routes can be generated. So, the best bounds may imply a higher computational cost than weaker bounds. This leads to a trade-off between the CPU time used in computing strong bounds and the size of the explored-tree that motivates the use of some standard strategies (Desaulniers et al., 2002) to improve the overall algorithmic performance.

In this way, to reduce the "tailing-off" effect which consists in a very low convergence-rate at the last iterations, the procedure ends after at most 10 iterations in no-root nodes, thus allowing a larger branching tree. Once that the nested column generation procedure is unable to provide cargo-routes, the node local lower bound (LLB) is computed and its integrality checked. With the purpose of improving the current GUB along the search tree, fast integer solutions are searched and provided by `GUROBI`, just in case the LLB is not integer. The node selection strategy is *best-first-search*; i.e. the node with the lowest LLB from the pool of unsolved subspaces is selected.

```
Loop(node,
    bestbound ← min_waiting(n) bound(n);
    current(n) ← no;
    current(waiting(n)):(bound(n) = bestbound)  ← yes;
    first ← 1;
    Loop(current: first,
        first ←  0;
        waiting(current) ←  no;
        Loop(i: i∈I⁺, loop(j: j∈I⁻,fx_ij ← no));
        Loop(i: i∈I⁺, loop(j: j∈I⁻ and (fx¹(current,i,j) or
fx⁰(current,i,j)), fx_i ← yes);
        Loop(r, exception(r) ← no;);
        Loop(i: i∈I⁺,
            Loop(j: j∈I⁻ and fx_ij and fx⁰(current,i,j),
                if(a_irt=1 and a_jrt=1, then exception(r) ← yes);
            );
            Loop(j: j∈I⁻ and fx_ij and fx¹(current,i,j),
                if(a_irt = 1 and a_jrt = 0,  then exception(r) ← yes);
                if(a_irt = 0 and a_jrt = 1,  then exception(r) ← yes);
            );
        );
    );
    While (eq. (10) < 0,
        Solve the linear RMP;
        Solve the mid-level route generation problem;
        Solve the patterns generation problems;
        Feed columns and patterns to the RMP;
    );
    LLB ← Obj. Function (linear RMP on the current node columns)
    Loop(i: i∈I⁻, π_i⁻ ← Duals from constraints (7));
    Loop(t, Loop(k,
          Loop(i: i∈I⁻,
              π^Max_itk- ← Duals from constraint (8.a);
              π^Min_itk- ← Duals from constraint (8.b);
          );
          Loop(i: i∈I⁺,
              π^Max_itk+ ← Duals from constraint (9.a);
              π^Min_itk+  ← Duals from constraint (9.b);
          );
    ); );
    If(LLB = integer,
       if (LLB < bestFound,
         bestFound ← LLB;
         Loop(t,
             Loop(r, bestX(r,t) ← X_rt;
                 Loop(i, best(r,i) ← a_irt;);
             );
         );
       );
    Else
       GUB ← Obj. Function (integer RMP on all columns)
       if (GUB < bestfound,
          bestfound ← GUB;
          Loop(t,
              Loop(r, bestX(r,t) ← X_rt;
                   Loop(i,best(r,i) ← a_irt));
               );
          );
       );
       Loop(i∈I⁺,
            Loop(j∈I⁻ and not fx_ij, π*_ij ← sum(pr,  π*_ip + π⁻_jp););
            Loop(j∈I⁻: max π*_ij and not fx_ij, fx_ij ← yes;);
       );
       first2 ← 1;
       Loop(fx:first2,
           first2 ← 0;
           newnode(n) ← newnode(n-1);
           fx⁰(newnode,i,j) ← fx⁰(current,i,j);
           fx¹(newnode,i,j) ← fx¹(current,i,j);
           bound(newnode) ← LLB;
           waiting(newnode) ← yes;
           fx⁰(newnode,fx) ← yes;

           newnode(n) ← newnode(n-1);
           fx⁰(newnode,i,j) ← fx⁰(current,i,j);
           fx¹(newnode,i,j) ← fx¹(current,I,j);
           bound(newnode) ← LLB;
           waiting(newnode) ← yes;
           fx¹(newnode,fx) ← yes;
       );
     );
   wait(n) ← no; wait(waiting) ← yes;
   waiting(wait):(bound(wait) > bestfound) ← no;
done:(card(waiting) = 0) = 1 );
```

Nodes processing Loop selecting the minimum bound waiting node.

Process the first node ***current***(*n*) *of the waiting list* ***waiting***(*n*).

Fix the branching pair (i,j).

Selects columns allocated to the ***current***(*n*) *node by excluding columns recorded in the list* ***exception***(*r*).

Solves the nested column generation procedure until no more cargo-routes can be generated.

Computes the LLB.
Update duals.

Update the best integer solution and record it in the list ***best***(*r*,*i*).

Selects the unfixed couple (i,j) with the maximum reduced cost to branch.

Generates two child nodes.

Terminate waiting nodes with LLB > bestFound.
Ends the nodes processing Loop.

Figure 4: Sketch of the branch-and price algorithm

The algorithm runs in a 2-cores 2.8-Ghz 16-Mbytes RAM PC and the mechanism settings used to solve the problems are summarized in Table 1.

Table 1: Setting options for the algorithm

Option	
MIP solver (routes generation problem)	CPLEX 11
MIP solver (DPs generation problem)	CPLEX 11
MIP solver (heuristic GUB computing problems)	Gurobi
Branching rule	On couples($i \in I^+, i \in I^-$)
Nodes selection strategy	Best first search
Maximum CPU time per master-slave iteration (s)	60
Multiple columns generated per iteration	Yes
Filtering of columns visiting the same subset of customers	Yes
Time-windows reduction and pre-processing	Yes
Maximum number of master-slave iterations per b&p node	20 (root)/10(no-root)
Maximum number of heuristic route generation iterations	10(root)
Maximum number of branch-and-price inspected nodes	10
Columns pool	Up to 10000 cargo-routes

5. Computational results

The solution procedure was first tested on a very small example for illustration purposes. Later, the algorithm was used to solve several instances generated from data of a case study presented by Marchetti et al. (2014). This is done with the aim of testing the capability of the procedure to reach good solutions for realistic-size problems with acceptable CPU times.

5.1 A small illustrative example

Due to the complex characteristics of the procedure, we first present and solve a very small example to illustrate the multiple issues involved in the distribution and inventorying of several products and also to show the output provided by the solution strategy. In this way, let us consider the delivery of two products (k_1 and k_2) from a single plant P to three customers, i_1, i_2 and i_3, over a planning period of three days (t_1, t_2 and t_3). Storage, production and consumption parameters for the plant and for the customers are summarized in Table 2. Cartesian coordinates that allow computing

distances, travelling costs and travelling times between customers and the plant are also presented in this table. The Figure 5 sketches the hypothetical and infeasible evolution of inventories on the plant and on the customers without any delivery trip. To avoid this infeasible evolution, some trucks with two compartments (c_1; c_2) of 7.5 units-capacity each-one must transport a quantity of products from the plant to the customers. The amount of product delivered to each customer is a decision left to the solution strategy. The algorithm ran to find the minimum cost solution meeting customer demands while satisfying inventorying constraints. The procedure generated, in addition to the 18 initialization cargo-routes, 6 more routes with its associated DPs and reported the solution specified in Table 3. Selected routes are illustrated in Figure 6 while the resulting inventories evolutions on the plant and on the customers are illustrated in Figure 7. The solution involves a single route during the first day, another route during the second day and two routes during the last day.

Table 2: Data for the small illustrative example

I	K	I_k^0	I_k^{Min}	I_k^{Max}	Consumption/production			(X, Y)
					t_1	t_2	t_3	Coordinates
P	k_1	13	0	15	3	3	3	$X = 2$
	k_2	13	0	15	3	3	3	$Y = 0$
i_1	k_1	3	0	6	2	2	3	$X = 0$
	k_2	-	-	-	-	-	-	$Y = 2$
i_2	k_1	-	-	-	-	-	-	$X = 2$
	k_2	4	0	5	3	2	3	$Y = 3$
i_3	k_1	3	0	5	3	3	3	$X = 4$
	k_2	4	0	5	3	3	3	$Y = 2$

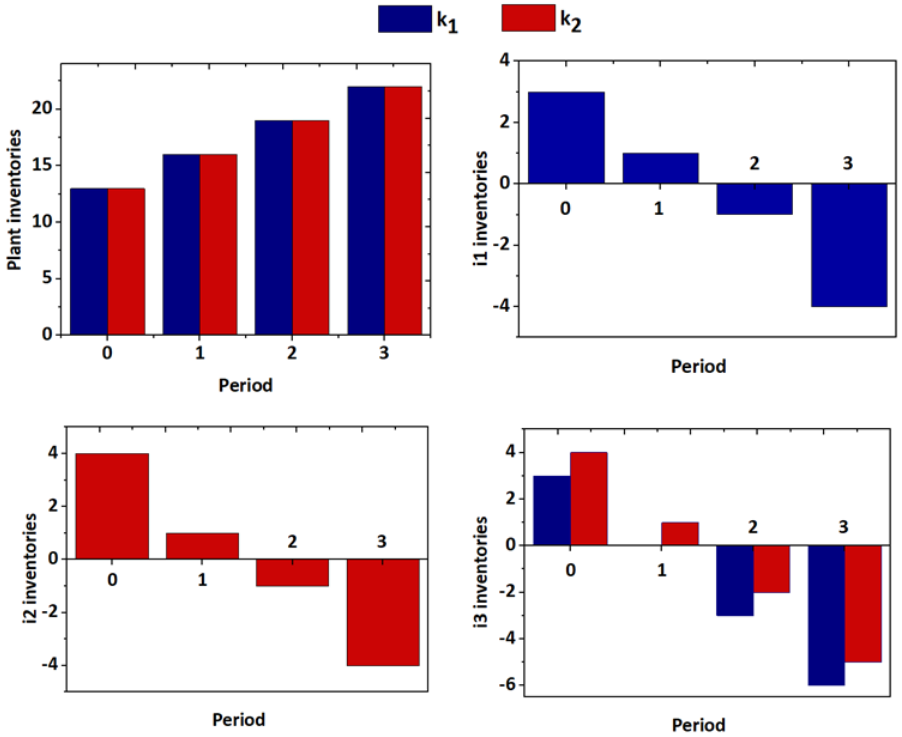

Figure 5: Hypothetical and infeasible evolution of inventories for the small illustrative example

Table 3: Routes and delivery schedules for the small illustrative example.

Route	Location	Arrival time	c_1 load	c_2 load
		Period 1		
1	P	--	k_2: +1	k_1: +1
	i_3	2.83	-	k_1: -1
	i_2	6.07	k_2: -1	-
	P	10.07	-	-
Routing time: 10.07		Routing cost: 8.07		
		Period 2		(k_1)
1	P	--	k_2: +4	k_1: +4
	i_3	2.83	k_2: -3	k_1: -3
	i_2	6.07	k_2: -1	-
	i_1	10.3	-	k_1: -1
	P	13.10	-	-
Routing time: 13.1		Routing cost: 10.1		
		Period 3		
1	P	--	k_2: +7	k_1: +6
	i_3	2.83	k_2: -4	k_1: -1
	i_2	6.07	k_2: -3	-
	i_1	10.3	-	k_1: -5
	P	13.10	-	-
Routing time: 13.1		Routing cost: 10.1		
2	P	--	k_2: +2	k_1: +5
	i_3	2.83	k_2: -1	k_1: -5
	i_2	6.07	k_2: -1	-
	P	10.07	-	-
Routing time: 10.07		Routing cost: 8.07		

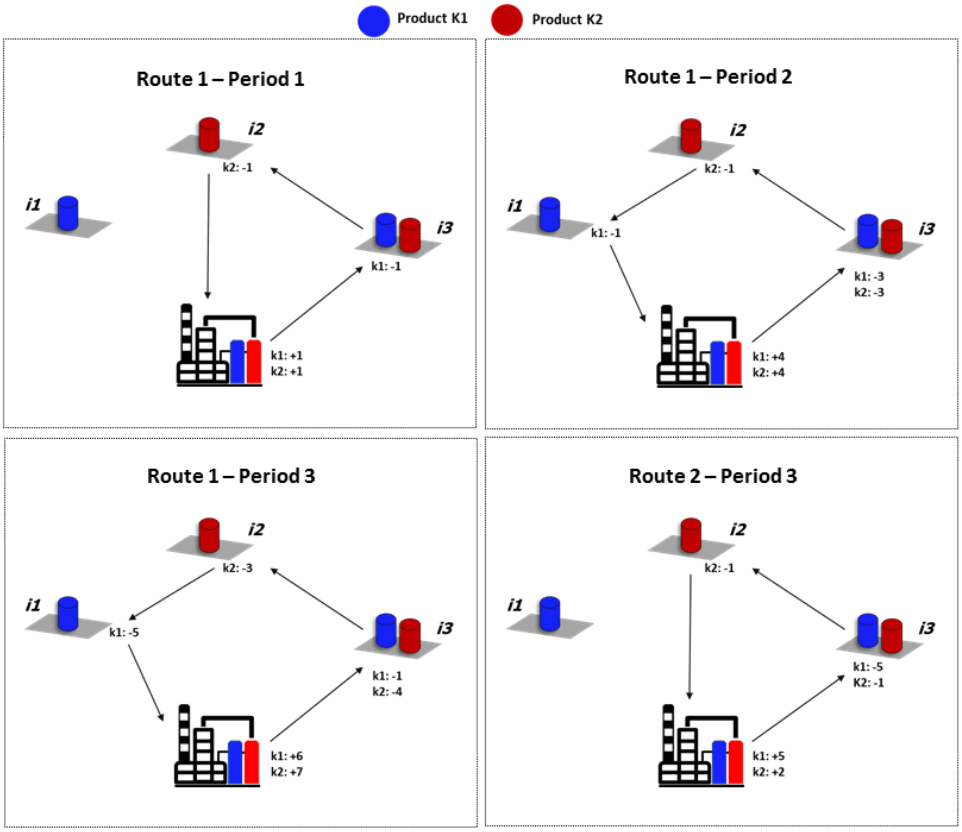

Figure 6: View of the routes for the small illustrative example

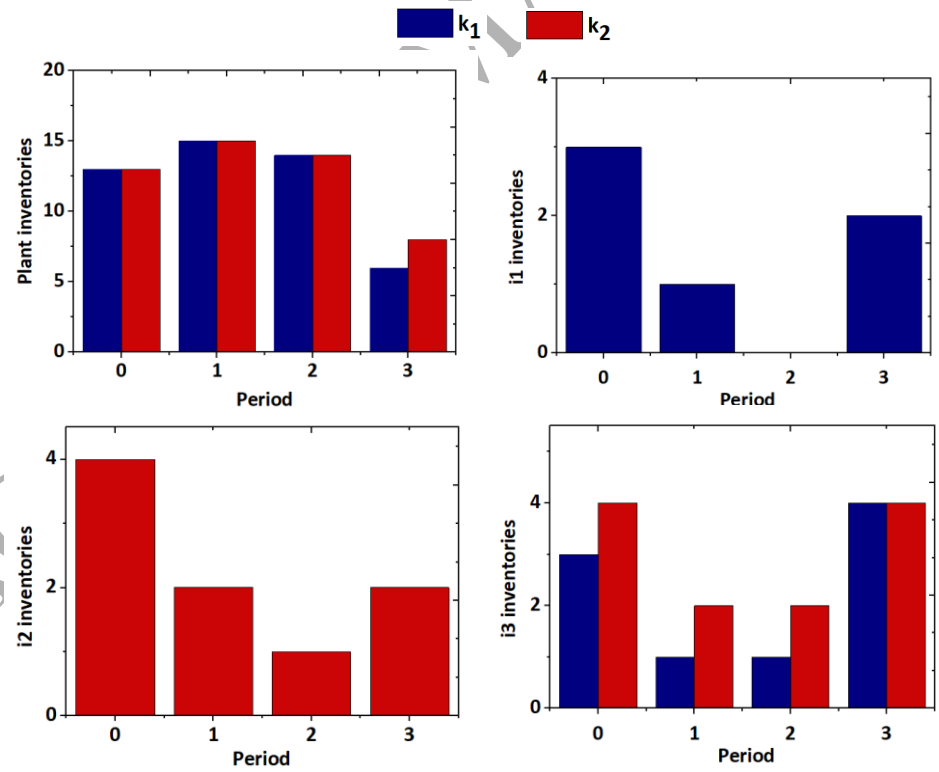

Figure 7: Evolution of inventories according to the found solution for the small illustrative example

5.2.Testing examples

In this section, the algorithm was tested on instances generated from data presented by Marchetti et al. (2014). The authors proposed an example involving the production and distribution of two products (LIN and LOX) from 3 plants in order to supply 50 customers over a time-horizon of up 14 days. Customers are randomly placed in different geographical sites and plants and customer locations are defined by the (X, Y) coordinates in the Euclidean plane (See figure 7). Distances (in miles) between plants and customers are computed from (X, Y) coordinates as Euclidean distances. The travelling times are computed from distances information by assuming an average speed of 40 miles/hour and constant loading/unloading times at plants and customers of st_i = 1 *h*.

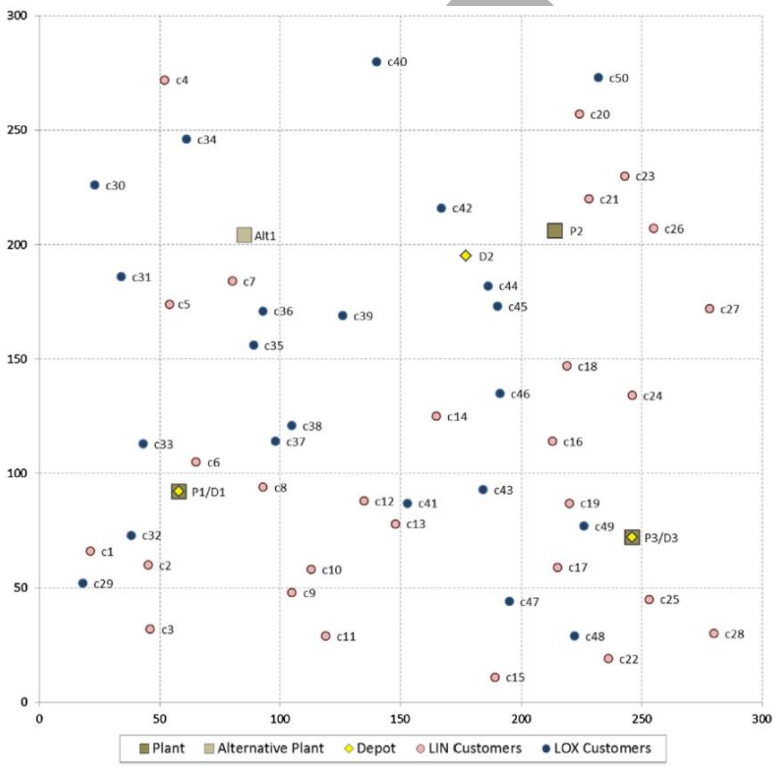

Figure 8: Geographical distribution of plants and customers (reprinted from Marchetti et al., 2014)

Instances data taken from Marchetti et al. (2014) can be downloaded from http://dx.doi.org/10.1016/j.compchemeng.2014.06.010. The parameter "redline" represents here the safety stock at the end of any time-period while "maximum" represents the maximum inventory at the beginning of any time-period. They are therefore conservative parameters. The plants host an unspecified number of vehicles with two compartments of capacity q_v = 630 ft^3. From these instances, variants with and without time windows were here generated by introducing the parameters reported in Table S1 presented as supplementary information. The solved instances involve the supply of both products from up to 3 plants. Such examples are defined by the number of LIN customers, the number of LOX customers, the number of time-periods of the planning horizon and the use (or not) of time-windows. Plants productions were adjusted to roughly meet the demand of the customers considered in each instances. Table 4 lists for each instance the following data: the production of LIN and LOX in the plants, the number of LIN' customers, the quantity of LOX' customers and the number of days of the planning horizon. Instances are named according to the number of serviced customers, the number of the supplying plants and the cardinality of the planning horizon. E.g. instance 12-1-14 refers to an instance of size-12 (i.e. $|I|_{LIN}$ = 7, $|I|_{LOX}$ = 5), a single supply plant and a planning period of 14 days. Solution parameters for the minimum-cost solutions to instances without time windows are presented in Table 5 while solution parameters for the minimum-cost solutions to instances with time windows are summarized in Table 6. Solution parameters are the objective function value IS, the number of generated columns, the total CPU time, the accumulated CPU time spent on the restricted master problem (RMP), the routes generation problem (RGP), the restricted pattern-generation master problem (RPGMP) and the pattern generation problem (PGP).

The conclusions that can be extracted from the information presented in both tables are the following: (i) instances with a given number of customers and plants involving a 14 days planning horizon are considerably harder to solve than the instances involving a 7 days planning horizon, except in the last instance (last two rows), which shows a reverse behavior. This is explained by the utilization of existing inventories on customers and the utilization of the remaining storage capacity on the plants. (ii) The bottleneck of the algorithm lies mostly on the RGP. The pattern generation stage involving both the RPGMP and the PMP seems to consume less time in large instances.

Table 4: Production data and number of customers for the testing instances

Instance	I^+								
	P_1		P_2		P_3		$\lvert I \rvert_{LIN}$	$\lvert I \rvert_{LOX}$	$\lvert T \rvert$
	LIN	LOX	LIN	LOX	LIN	LOX			
12-1-7	750	350	-	-	-	-	7	5	7
12-1-14	750	350	-	-	-	-	7	5	14
25-1-7	1800	900	-	-	-	-	14	11	7
25-1-14	1800	900	-	-	-	-	14	11	14
12-2-7	500	250	300	150	-	-	7	5	7
12-2-14	500	250	400	200	-	-	7	5	14
25-2-7	1200	600	600	300	-	-	14	11	7
25-2-14	1200	600	600	300	-	-	14	11	14
37-2-7	1500	750	1000	500	-	-	21	16	7
37-2-14	1500	750	1000	500	-	-	21	16	14
50-2-7	2000	1000	1500	750	-	-	28	22	7
50-2-14	2000	1000	1500	750	-	-	28	22	14
50-3-7	1500	800	1500	750	500	500	28	22	7
50-3-14	1500	800	1500	750	400	400	28	22	14

Table 5: Solution data for large scale instances without time windows

Instances	Solution data		CPU time				
	IS ($)	Columns	RMP (*s*)	RGP (*s*)	RPGMP (*s*)	PGP (*s*)	Total (*s*)
12-1-7	221.4	77	0.4	8.7	3.5	23.1	35.8
12-1-14	855.7	266	1.7	13.1	11.1	99.8	125.6
25-1-7	405.4	210	1.9	731.1	12.6	84.4	827.0
25-1-14	855.7	68	1.7	13.0	11.6	96.6	122.9
12-2-7	221.8	70	0.5	5.6	2.6	17.8	26.5
12-2-14	911.1	140	1.1	6.4	4.8	40.9	53.2
25-2-7	409.3	322	2.4	723.6	12.7	107.3	846.0
25-2-14	2050.1	672	5.1	694.7	28.2	259.0	987.0
37-2-7	639.4	413	3.0	1535.6	16.5	137.8	1692.9
37-2-14	3531.9	812	5.9	1518.6	26.8	273.1	1824.5
50-2-7	1002.9	414	3.4	1808.4	16.4	137.4	1965.6
50-2-14	3222.1	812	6.1	1806.7	28.9	228.2	2069.9
50-3-7	894.7	623	5.9	2710.4	36.0	248.9	3001.2
50-3-14	2640.7	770	8.5	2346.7	40.8	276.5	2672.6

Table 6: Solution data for large scale instances with time windows

Instances	Solution data		CPU time				
	IS ($)	Columns	RMP (*s*)	RGP (*s*)	RPGMP (*s*)	PGP (*s*)	Total (*s*)
12-1-7-TW	254.2	112	0.6	4.5	3.8	39.1	48.0
12-1-14-TW	924.5	196	1.6	3.9	7.3	63.5	76.3
25-1-7-TW	441.6	210	1.8	19.6	11.8	80.6	114.8
25-1-14-TW	1091.9	1190	1.2	21.1	80.6	439.8	1069.2
12-2-7-TW	254.3	126	0.9	2.4	5.4	44.9	53.5
12-2-14-TW	934.8	182	1.2	3.0	8.8	68.1	81.1
25-2-7-TW	447.6	210	2.0	19.7	11.2	77.2	110.3
25-2-14-TW	2376.9	392	4.0	23.5	24.1	169.0	220.6
37-2-7-TW	740.6	211	2.2	126.6	11.3	78.2	218.3
37-2-14-TW	4005.1	420	4.4	129.5	20.7	146.7	301.3
50-2-7-TW	1196.9	220	2.4	281.8	10.1	69.4	363.7
50-2-14-TW	5912.8	423	5.1	380.6	20.7	141.7	548.2
50-3-7-TW	943.4	210	3.3	340.1	11.7	83.2	438.4
50-3-14-TW	4944.7	420	6.1	331.7	23.0	162.0	521.8

From Table 5, it follows that instance 50-3-7 consumed more CPU resources than instance 50-3-14. This anomaly may be originated from a slow convergence rate of the PGP problem of instance 50-3-7. Note that CPU times consumed in RGP, RPGMP and PGP problems are consistent with the pattern observed in remaining instances. The anomaly is also observed by comparing CPU times of the RGP stage of instance 50-3-7-TW with the RGP stage of instance 50-3-14-TW.

In Table S2, presented as supplementary information, we detail the solution provided to example $|I|_{\text{LIN}} = 7$; $|I|_{\text{LOX}} = 5$; $|T| = 7$ with a single plant producing 750 ft^3 of LIN and 350 ft^3 of LOX. Such a table details the routes corresponding to each day of the planning horizon and the evolution of inventories both in the plant and in the customers. It seems that the number of routes used on each period grows as the time goes on. This pattern, which was also observed in other unreported instances, seems to indicate that products inventories and idle storage capacities are used as far as possible to save routing costs. As the allocation of products to compartments is in itself another combinatorial problem, the number of DPs per route depends mainly on two factors: the number of customers visited along the route and the assignment of products to

compartments. In spite of this, it seems that CPU spent on PGPs grows, with instances sizes, at a slower rate than the growth of CPU times spent on RGPs. This is because scheduling decisions, necessary to sequence clients along a route, are not involved in the patterns generation phase. This is another reason to decouple routing decisions from delivering decisions.

6. Conclusions

This work developed a decomposition strategy based on a nested CG procedure for planning, over a multi-period time horizon, the distribution and inventorying of several chemical fluids. Products demands and plants productions were assumed to be known data after statistical forecasting.

This one is the second work of a research line aimed at the optimal integration of inventorying and delivery of chemical fluids. The procedure is used for designing over a multi-period time-horizon the best routes for distributing multiple chemical fluids from plants to customers. Moreover, the solution indicates the time to serve a given customer, the quantity of products to deliver to the visited customers and the optimal sequence of visited customers by each vehicle-route on each time period. The procedure is able also to consider the allocation of several products on the same vehicle by fixing the quantity and type of product transported on each compartment. The freedom to select the delivery-period and to fix the quantity of products to deliver to each customer allows saving transportation costs through a very efficient use of the compartments of the vehicles, inventories and idle storage capacities. For developing the decomposition strategy, an inventorying-routing problem tailored to the distribution of chemical fluids was first modeled as a set partitioning problem with additional balances constraints over the whole planning horizon for each product.

Routes and delivery patterns were separately computed through a nested CG mechanism on each node of an incomplete branch-and-price tree. The goal was to decouple routing decisions from delivering decisions in order to avoid convergence problems happening in a conventional column generation algorithm. The proposed mechanism has been used to solve numerous instances featuring two different planning-horizons and several numbers of customers and plants. In all examples, the solutions were obtained with moderate computational effort.

The next step of this research line, to consider in a future work, aims at dealing with service times depending both on the cargo to load and unload and to take into account the delivery-time within a time-period for getting less conservative solutions. Inventorying costs on customers, additional restrictions to consider stability issues on vehicles as well as modulation of production levels on plants should also be taken into account in order to research inventorying and distribution of perishable products.

Acknowledgements

This work was supported by grants 50120110100315LI from Universidad Nacional del Litoral; grant BID-PICT 2392 from the Agencia Nacional de Promoción Científica y Tecnológica and grant PIP 112 20150100641 from Consejo Nacional de Investigaciones Científicas y Técnicas.

References

Adulyasak, Y., Cordeau, J.-F., Jans, R. 2015.The Production Routing Problem: A Review of Formulations and Solution Algorithms. *Comput. Oper. Res.* 55 (C), 141-152.

Aksen, D., Kaya, O., Salman, F., Akça, Y. 2012. Selective and periodic inventory routing problem for waste vegetable oil collection. *Optim. Lett.* 6(6):1063–1080.

Alegre, J., Laguna, M., Pacheco, J. 2007. Optimizing the periodic pickup of raw materials for a manufacturer of auto parts. *Eur. J. Oper. Res.* 179(3), 736–746.

Bard, J., Huang, L., Jaillet, P., Dror, M. 1998. A decomposition approach to the inventory routing problem with satellite facilities. *Transp. Sci.* 32(2):189–203.

Bell, W.J., Dalberto, L.M., Fisher, M.L., Greenfield, A.J., Jaikumar, R., Kedia P., Mack, R.G., Prutzman P.J. 1983. Improving the distribution of industrial gases with an on-line computerized routing and scheduling optimizer. *Interfaces* 13(6):4–23.

Blumenfeld, D.E., Burns, L.D., Diltz, J.D., Daganzo, C.F. 1985. Analyzing trade-offs between transportation, inventory and production costs on freight networks. *Transp. Res. Part B.* 19(5), 361–380.

Campbell, A.M., Savelsbergh, M.W.P. 2004 A decomposition approach for the inventory-routing problem. *Transp. Sci.* 38(4):488–502.

Christiansen, M., Fagerholt, K., Flatberg, T., Haugen, Ø., Kloster, O,, Lund, E.H. 2011. Maritime inventory routing with multiple products: A case study from the cement industry. *Eur. J. Oper. Res.* 208(1), 86–94.

Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D. 2007. Maritime transportation. Barnhart C, Laporte G, eds. Transportation, Handbooks in *Oper. Res. and Man. Sci.*, 14 (North-Holland, Amsterdam), 189–284.

Christiansen, M., Fagerholt, K., Nygreen, B., Ronen, D. 2013. Ship routing and scheduling in the new millennium. *Eur. J. Oper. Res.* 228(3), 467–483.

Christiansen, M., Fagerholt, K., Ronen. D. 2004. Ship routing and scheduling: Status and perspectives. *Transp. Sci.* 38(1), 1–18.

Cóccola, M., Méndez, C., Dondo R. 2017. A decomposition framework for managing inventory and distribution of fluid products by an order-based-resupply methodology. *Comput. Chem. Eng.*, 106, 373-384.

CPLEX 12 Solver Manual. GAMS. The Solver Manuals, (2007).

Custódio, A., Oliveira, R. 2006. Redesigning distribution operations: A case study on integrating inventory management and vehicle routes design. *Int. J. Logist*., 9(2), 169–187.

Desaulniers, G. 2010. Branch-and-price-and-cut for the Split-Delivery Vehicle Routing Problem with Time Windows. *Oper. Res.* 58 (1), 179-192.

Desaulniers, G., Desrosiers, J., Solomon M. 2002. Accelerating Strategies in Column Generation Methods for Vehicle Routing and Crew Scheduling Problems. Essays and Surveys in Metaheuristics. *Oper. Res./Comput. Sci. Interfaces Series*, 15, 309-324.

Desaulniers, G., Rakke, J.G., Coelho, L. 2016. A Branch-Price-and-Cut Algorithm for the Inventory-Routing Problem. *Transp. Sci*., 50 (3), 1060-1076.

Dong, Y., Maravelias, C.T., Pinto, J.M., Sundaramoorthy, A. 2017. Solution methods for vehicle-based inventory routing problems. *Comput. Chem. Eng*., 101, 259–278.

Dror, M., Ball, M., Golden, B. 1985. A computational comparison of algorithms for the inventory routing problem. *Annals Oper. Res*., 4, 3-23.

Federgruen, A., Prastacos, G., Zipkin, P.H. 1986. An allocation and distribution model for perishable products. *Oper. Res*. 34(1), 75–82.

Federgruen, A., Zipkin, P. 1984. A combined vehicle routing and inventory allocation problem. *Oper. Res.* 32(5), 1019–1037.

Gaur, V., Fisher, M.L. 2004. A periodic inventory routing problem at a supermarket chain. *Oper. Res*. 52(6), 813–822.

Golden, B.L., Assad, A.A., Dahl, R. 1984. Analysis of a large-scale vehicle-routing problem with an inventory component. Large Scale Systems Inform. *Decision Tech*. 7(2–3), 181–190.

Grossmann, I. 2012. Advances in mathematical programming models for enterprise-wide optimization. *Comput. Chem. Eng.*, 47, 2-18.

Hemmelmayr, V., Doerner, K.F., Hartl, R.F., Savelsbergh, M.W.P. 2009. Delivery strategies for blood products supplies. *OR Spectrum*, 31(4), 707–725.

Hennig, F., Nygreen B., Lübbecke, M. 2012. Nested Column Generation applied to the Crude Oil Tanker Routing and Scheduling Problem with Split Pickup and Split Delivery. *Naval Research Logistics*, 59 (3-4), 298-310.

Marchetti, P., Gupta, V., Grossmann, I., Cook, L., Valton, P., Singh, T. 2014. Simultaneous production and distribution of industrial gas supply-chains. *Comput. Chem.. Eng.*, 69, 39–58.

Mercer, A., Tao, X. 1996. Alternative inventory and distribution policies of a food manufacturer. *J. Oper. Res. Soc*. 47(6), 755–765.

Oppen, J., Løkketangen, A., Desrosiers, J. 2010. Solving a rich vehicle routing and inventory problem using column generation. *Comput. Oper. Res.* 37(7), 1308–1317.

Popovic, D., Vidovic, M., Radivojevic, G. 2012. Variable neighborhood search heuristic for the inventory routing problem in fuel delivery. *Expert Systems Appl.* 39(18), 13390–13398.

Ronen, D. 1993. Ship scheduling: The last decade. *Eur. J. Oper. Res*. 71(3):325–333.

Savelsbergh, M., Sol, M. 1998. Drive: Dynamic Routing of Independent Vehicles *Oper. Res.*, 46 (4), 474-490.

Shen, Q., Chu, F., Chen, H. 2011. A Lagrangian relaxation approach for a multi-mode inventory routing problem with transshipment in crude oil transportation. *Comput. Chem Eng.* 35 (10), 2113-2123.

Singh, T., Arbogast, J., Neagu, N. 2015. An incremental approach using local-search heuristic for inventory routing problem in industrial gases. *Comput. Chem. Eng.*, 80, 199–210.

Stacey, J., Natarajarathinam, M., Sox, C. 2007. The storage constrained, inbound inventory routing problem. *Int. J. Phys. Distribution Logist Manag*. 37(6):484–500.

Trudeau, P., Dror, M. 1992. Stochastic inventory routing: Route design with stockouts and route failures. *Transportation Sci*. 26(3), 171–184.

You, F., Pinto, J., Capón, E., Grossmann, I,; Arora, N., Megan, L. 2011. Optimal distribution-inventory planning of industrial gases: I. Fast computational strategies or large-scale problems. *Ind. Eng. Chem. Res.*, 50, 910–2927.

Zamarripa, M, Marchetti, P., Grossmann, I., Singh, T, Lotero, I., Gopalakrishnan, A., Besancon, B., André, J. 2016. Rolling Horizon Approach for Production−Distribution Coordination of Industrial Gases Supply Chains. *Ind. Eng. Chem. Res.*, 55, 2646−2660.

Appendix A

Minimum number of visits to customers and production plants. The minimum number of visits to a given customer along the whole planning period can be computed from the following expression:

$$v_i = \left\lceil \frac{1}{|C|} \sum_{k \in K} \left\lceil \frac{\max\left(\sum_{t \in T} d_{itk} - \left(I_{ik}^{0} - I_{ik}^{\min} \right); 0 \right)}{q_c} \right\rceil \right\rceil \quad \forall i \in I^{-} \tag{A.1}$$

In the same way, the minimum number of vehicles used for evacuation the production from each plant can be computed from this expression:

$$v_i = \left\lceil \frac{1}{|C|} \sum_{k \in K} \left\lceil \frac{\max\left(\sum_{t \in T} p_{itk} - \left(I_{ik}^{\max} - I_{ik}^{0} \right); 0 \right)}{q_c} \right\rceil \right\rceil \quad \forall i \in I^{+} \tag{A.2}$$

Upper bound on the quantity to pick up/deliver from/to source/sink nodes. The maximum quantity of product *k* that can be picked-up (delivered) from each plant (customer) is calculated according to the following expressions:

$$q_{itk} = \min\left(\left(I_{ik}^{\max} - I_{ik}^{0} + \sum_{\substack{t' \in T: \\ t' \le t}} d_{it'k} \right); \left(I_{ik}^{\max} - I_{ik}^{\min} \right); |C| q_c \right) \quad \forall t \in T, i \in I^{-}, k \in K : \sum_{\substack{t' \in T: \\ t' \le t}} d_{it'k} > 0 \tag{A.3}$$

$$q_{itk} = \min\left(\left(I_{ik}^{0} - I_{ik}^{\min} + \sum_{\substack{t' \in T: \\ t' \le t}} p_{it'k} \right); \left(I_{ik}^{\max} - I_{ik}^{\min} \right); |C| q_c \right) \quad \forall t \in T, i \in I^{+}, k \in K : \sum_{\substack{t' \in T: \\ t' \le t}} p_{it'k} > 0 \tag{A.4}$$

Graphical Abstract